Package: dipca (via r-universe)

June 4, 2026

Type Package

Title Dynamic Inner Principal Component Analysis

Version 0.1.0

Description Implements Dynamic Inner Principal Component Analysis (DiPCA), Dynamic Inner Canonical Correlation Analysis (DiCCA), and Dynamic Inner Partial Least Squares (DiPLS) for multivariate time series analysis. These methods extract latent components that follow autoregressive models, enabling temporal prediction and reconstruction. Based on the methods of Dong and Qin (2018) <doi:10.1016/j.jprocont.2017.05.002> for DiPCA, Dong and Qin (2018) <doi:10.1016/j.ifacol.2018.09.379> for DiCCA, and Dong and Qin (2018) <doi:10.1016/j.jprocont.2018.04.006> for DiPLS. Integrates with the 'multivarious' package framework for consistent preprocessing and projection interfaces.

License GPL-3

Encoding UTF-8

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.0), multivarious, stats, forecast

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0), knitr, rmarkdown, ggplot2, gridExtra, reshape2

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs make

Repository https://bbuchsbaum.r-universe.dev

Date/Publication 2026-02-14 20:45:14 UTC

RemoteUrl https://github.com/bbuchsbaum/dipca

RemoteRef HEAD

RemoteSha 7fc174f62911a97ad15aa415ad1918d6fb7502e5

Contents

`arima_dicca` *Compact DiCCA with AR/ARMA/ARIMA inner models (measurement-space update)*

Description

Implements the measurement-space DiCCA alternating scheme with an inner AR/ARMA/ARIMA model fitted on each component. The deflation step follows Dong & Qin (2018): $X \leftarrow X - tp^\top$ with $p = X^\top t/(t^\top t)$.

Usage

```
arima_dicca(
  X,
  n_comp,
  burnin = 0,
  center = TRUE,
  scale = FALSE,
  inner = c("arma", "arima", "ar"),
  order_mode = c("select_once", "fixed", "auto_each"),
  max_iter = 200L,
  max_iter_one = 100L,
  tol = 1e-06,
  order = NULL,
  pmax = 5L,
  qmax = 5L,
  verbose = 1L,
  fit_every = Inf,
  experimental = FALSE
)
```

Arguments

X	numeric matrix (T x m), rows are time, columns variables.
n_comp	number of dynamic latent variables to extract.
burnin	integer lag order s (>=1). Historically the compact routine used a burn-in; we treat this as the lag order for consistency with the main dicca() function.
center	logical; center columns before fitting.
scale	logical; scale to unit variance if TRUE.
inner	one of "ar", "arma", or "arima" (experimental; see experimental argument).
order_mode	how to handle ARIMA orders across iterations: "select_once" (default) selects once per component and reuses; "fixed" requires the order argument; "auto_each" re-selects at every iteration.
max_iter	integer maximum outer iterations for each component.
max_iter_one	retained for backwards compatibility; the effective iteration cap is min(max_iter, max_iter_one).
tol	numeric convergence tolerance on the objective.
order	optional list(p=, d=, q=) when order_mode = "fixed".
pmax, qmax	upper bounds when auto-selecting orders.
verbose	integer verbosity (>=0).
fit_every	integer; refit the inner model every fit_every iterations (default: Inf, fit only once).
experimental	logical; when TRUE enables the experimental inner = "arima" pathway, otherwise requesting "arima" will error unless the option options(dipca.experimental_arima = TRUE) has been set.

Value

a list with fields: W, P, T, models, info, R, method; with class 'dicca_model'.

---

dicca *Dynamic-Inner Canonical Correlation Analysis (DiCCA) Extract dynamic latent variables by maximizing correlation between each latent series and its AR(s) prediction (canonical correlation form). This follows Dong & Qin (2018 IFAC) iteration and deflation.*

Description

Dynamic-Inner Canonical Correlation Analysis (DiCCA) Extract dynamic latent variables by maximizing correlation between each latent series and its AR(s) prediction (canonical correlation form). This follows Dong & Qin (2018 IFAC) iteration and deflation.

Usage

```
dicca(
  X,
  s,
  l,
  preproc = multivarious::center(),
  tol = 1e-07,
  max_iter = 1000,
  n_init = 4,
  verbose = 0,
  seed = NULL,
  inner = c("classic", "ar", "arma", "arima")
)

DiCCA(
  X,
  s,
  l,
  preproc = multivarious::center(),
  tol = 1e-07,
  max_iter = 1000,
  n_init = 4,
  verbose = 0,
  seed = NULL,
  inner = c("classic", "ar", "arma", "arima")
)
```

Arguments

X	numeric matrix (T x m), time by variables.
s	integer lag order (>=1).
l	integer number of components to extract.
preproc	a pre_processor object from the multivarious package. Default is center() which centers columns. Use prep(pass()) for no preprocessing.
tol	numeric outer tolerance (‖w_{k+1}-w_k‖_inf).
max_iter	integer max outer iterations per component.
n_init	integer random restarts per component; best objective kept.
verbose	integer verbosity (0=quiet).
seed	optional RNG seed.
inner	one of "classic" (DiCCA coordinate updates), or compact variants "ar", "arma", "arima" (the latter is experimental and requires options(dipca.experimental_arima = TRUE)) which dispatch to the compact ARIMA-DiCCA routine.

Details

One-component iteration: see DiCCA Eqs. (4)–(8): beta = (T_s^T T_s)^{-1} T_s^T t_{s+1}; beta <- beta / sqrt(t_{s+1}^T T_s beta); X_beta = sum_i beta_i X_{s+1-i}; w <- (X_{s+1}^T X_{s+1}

+ X_beta^T X_beta)^+ [X_{s+1}^T (T_s beta) + X_beta^T t_{s+1}], then normalize w. After each component, deflate X by X <- X - t p^T with p = X^T t / (t^T t). A joint VAR(s) is fit on the extracted score matrix for prediction utilities.

Value

A `bi_projector` object of class `dicca` with fields:

- `v` - weight matrix (m x l)
- `s` - score matrix (T x l)
- `sdev` - standard deviations of components
- `preproc` - fitted preprocessor
- `loadings` - loading matrix (m x l)
- `betas` - beta coefficients (l x s)
- `theta` - VAR coefficients ((l*s) x l) [classic only]
- `R2` - R^2 values per component
- `lag_order` - lag parameter s
- `obj_history` - convergence history
- `iters_per_component` - iterations per component
- `compact_models` - per-component ARIMA models (compact variants only)
- `compact_info` - compact model metadata (compact variants only)

References

Dong & Qin (2018) Dynamic-Inner Canonical Correlation and Causality Analysis for High Dimensional Time Series Data, IFAC ADCHEM.

Examples

```
## Not run:
library(multivarious)
library(dipca)

# Simulate time series data
set.seed(123)
T <- 400; m <- 10; l <- 3; s <- 1
X <- matrix(rnorm(T * m), T, m)

# Fit DiCCA with default centering
fit <- dicca(X, s = 1, l = 3, n_init = 2, max_iter = 500)

# Fit with centering and scaling
fit_scaled <- dicca(X, s = 1, l = 3,
                    preproc = center() %>% colscale(type = "z"),
                    n_init = 2)

# Access components using bi_projector methods
```

```
scores <- scores(fit)
weights <- components(fit)
r2_values <- fit$R2

# Temporal prediction
predictions <- predict(fit, X)

## End(Not run)
```

`dicca_predict_scores` *One-step-ahead prediction of DLVs (diagonal G(B))*

Description

One-step-ahead prediction of DLVs (diagonal G(B))

Usage

```
dicca_predict_scores(model, scores)
```

Arguments

`model`	a fitted DiCCA model object with compact inner models
`scores`	numeric matrix of observed scores to predict from

Value

matrix of predicted scores

`dicca_scores` *Transform new data to Di(L)V scores using a fitted compact DiCCA model*

Description

Transform new data to Di(L)V scores using a fitted compact DiCCA model

Usage

```
dicca_scores(model, Xnew)
```

Arguments

`model`	a fitted DiCCA model object
`Xnew`	numeric matrix of new data to transform

Value

matrix of scores

dipca *Dynamic Inner PCA (DiPCA)*

Description

Fit dynamic inner principal components using the fast coordinate-maximization algorithm with optional inner power iterations for the w-update.

Usage

```
dipca(
  X,
  s,
  l,
  preproc = multivarious::center(),
  tol = 1e-07,
  max_iter = 1000,
  n_init = 4,
  algorithm = c("I", "II"),
  inner_power = 50,
  inner_tol = 1e-08,
  verbose = 0,
  seed = NULL
)

DiPCA(
  X,
  s,
  l,
  preproc = multivarious::center(),
  tol = 1e-07,
  max_iter = 1000,
  n_init = 4,
  algorithm = c("I", "II"),
  inner_power = 50,
  inner_tol = 1e-08,
  verbose = 0,
  seed = NULL
)

dipca_fit(
  X,
  s,
  l,
  preproc = multivarious::center(),
  tol = 1e-07,
  max_iter = 1000,
```

```
  n_init = 4,
  algorithm = c("I", "II"),
  inner_power = 50,
  inner_tol = 1e-08,
  verbose = 0,
  seed = NULL
)
```

Arguments

X	numeric matrix of shape (T, m), time by variables.
s	integer, lag order (>=1).
l	integer, number of components to extract.
preproc	a pre_processor object from the multivarious package. Default is center() which centers columns. Use prep(pass()) for no preprocessing.
tol	numeric, outer stopping tolerance on the eigen residual $\|\|d - \lambda w\|\|_\infty$.
max_iter	integer, maximum outer iterations.
n_init	integer, random restarts per component.
algorithm	"I" (one power step) or "II" (inner power iterations).
inner_power	integer, max inner power iterations (algorithm "II").
inner_tol	numeric, tolerance for inner power iteration.
verbose	integer, verbosity (0=quiet).
seed	optional integer RNG seed for reproducibility.

Details

The implementation follows the DiPCA objective and iteration given in Dong & Qin (2018, Table 2; Eqs 8–18) and the coordinate-maximization interpretation and residual criterion $||d - \lambda w||_\infty$ in Shin et al. (2020). The code avoids forming Y_i explicitly; all updates are realized via matvecs with lagged blocks X_{s+1-i}.

Value

A bi_projector object of class dipca with fields:

- v - weight matrix (m x l)
- s - score matrix (T x l)
- sdev - standard deviations of components
- preproc - fitted preprocessor
- loadings - loading matrix (m x l)
- betas - beta coefficients (l x s)
- theta - VAR coefficients ((l*s) x l)
- lag_order - lag parameter s
- obj_history - convergence history
- iters_per_component - iterations per component

Examples

```
## Not run:
library(multivarious)
library(dipca)

set.seed(1)
T <- 600; m <- 5; l <- 3; s <- 1

# Simulate VAR(1) latent process
A <- matrix(c(0.5205, 0.1022, 0.0599,
              0.5367, -0.0139, 0.4159,
              0.0412, 0.6054, 0.3874), 3, 3, byrow = TRUE)
P <- matrix(c(0.4316, 0.1723, -0.0574,
              0.1202, -0.1463, 0.5348,
              0.2483, 0.1982, 0.4797,
              0.1151, 0.1557, 0.3739,
              0.2258, 0.5461, -0.0424), 5, 3, byrow = TRUE)
t <- matrix(0, T, l)
v <- matrix(rnorm(T * l), T, l)
for (k in 2:T) {
  t[k, ] <- c(0.5205, 0.5367, 0.0412) + A %*% t[k - 1, ] + v[k, ]
}
X <- t %*% t(P) + matrix(rnorm(T * m, sd = 0.1), T, m)

# Fit DiPCA with centering (default)
fit <- dipca(X, s = 1, l = 3, n_init = 3, max_iter = 800,
             tol = 1e-7, algorithm = "I")

# Fit with centering and scaling
fit_scaled <- dipca(X, s = 1, l = 3,
                    preproc = center() %>% colscale(type = "z"),
                    n_init = 3, max_iter = 800, tol = 1e-7)

# Access components using bi_projector methods
scores <- scores(fit)           # Extract latent scores
weights <- components(fit)      # Extract weight matrix
loadings <- fit$loadings        # Extract loadings
theta <- fit$theta              # VAR coefficients

# Project new data
X_new <- matrix(rnorm(50 * m), 50, m)
scores_new <- project(fit, X_new)

# Reconstruct data
X_recon <- reconstruct_new(fit, X_new)

# Temporal prediction
predictions <- predict(fit, X_new)
str(predictions)  # scores and scores_hat

# Compute residuals
resid <- residuals(fit, X_new)
```

```
str(resid)  # v (score residuals), e_hat (data residuals), scores

## End(Not run)
```

`dipls` *Dynamic-inner Partial Least Squares (DiPLS)*

Description

Fit the Dynamic-inner PLS algorithm of Dong & Qin (2018) to jointly model input block X_t and output block Y_t. The outer loop updates the dynamic weights w, q, and FIR coefficients β following Appendix A of the paper, while the inner model is estimated either as a FIR filter (default) or an ARX refinement.

Usage

```
dipls(
  X,
  Y,
  n_comp,
  s,
  preproc_x = multivarious::center(),
  preproc_y = NULL,
  mode = c("fir", "arx"),
  max_iter = 200L,
  tol = 1e-07,
  verbose = 1L
)
```

Arguments

`X`	numeric matrix with T rows (time) and m columns (inputs).
`Y`	numeric matrix with T rows and p columns (outputs).
`n_comp`	integer number of dynamic latent components to extract.
`s`	non-negative integer lag order for the inner model.
`preproc_x`	a `pre_processor` object from the multivarious package for the X block. Default is `center()` which centers columns. Use `prep(pass())` for no preprocessing.
`preproc_y`	a `pre_processor` object from the multivarious package for the Y block. If NULL (default), uses the same preprocessing as `preproc_x`.
`mode`	character, either `"fir"` for the basic DiPLS inner model or `"arx"` to augment with AR terms on u.
`max_iter`	integer, maximum outer iterations per component.
`tol`	numeric tolerance on the maximum change of w, q, and β between outer iterations.
`verbose`	integer verbosity level (0 = quiet, 1 = per-component banner, >1 also logs iteration deltas).

Value

A `cross_projector` object of class `dipls` with fields:

- `vx` - X-block weight matrix (m x n_comp)
- `vy` - Y-block weight matrix (p x n_comp)
- `preproc_x` - fitted X-block preprocessor
- `preproc_y` - fitted Y-block preprocessor
- `T` - X-block score matrix (T x n_comp)
- `U` - Y-block score matrix (T x n_comp), first s rows are NA
- `P` - X-block loadings (m x n_comp)
- `C` - Y-block loadings (p x n_comp)
- `inner` - per-component inner model coefficients (list)
- `R` - X-block projection matrix
- `lag_order` - lag parameter s
- `mode` - inner model mode ("fir" or "arx")
- `iterations` - iteration counts per component

Examples

```
## Not run:
set.seed(1)
Tn <- 400; m <- 6; p <- 2; s <- 2
X <- matrix(rnorm(Tn * m), Tn, m)
w0 <- rnorm(m); w0 <- w0 / sqrt(sum(w0^2))
t0 <- as.numeric(X %*% w0)
beta0 <- c(0.8, -0.3, 0.2)
u <- numeric(Tn)
u[(s + 1):Tn] <- as.numeric(dipls_build_Ts_cpp(t0, s) %*% beta0)
Q0 <- matrix(rnorm(p), p); Q0 <- Q0 / sqrt(sum(Q0^2))
Y <- u %o% as.numeric(Q0) + matrix(rnorm(Tn * p, sd = 0.2), Tn, p)

fit <- dipls(X, Y, n_comp = 1, s = s, mode = "fir",
             preproc_x = multivarious::center(), verbose = 1)
Yhat <- predict(fit, X)

## End(Not run)
```

`dipls_scores` *Project X to DiPLS scores*

Description

Project X to DiPLS scores

Usage

```
dipls_scores(model, Xnew)
```

Arguments

model	fitted DiPLS object from dipls.
Xnew	numeric matrix with the same number of columns as the training X.

Value

Matrix of latent scores (rows correspond to observations).

---

predict.dicca *Predict Method for DiCCA*

Description

Temporal forecasting using the VAR model (classic) or per-component ARIMA models (compact).

Usage

```
## S3 method for class 'dicca'
predict(object, newdata = NULL, ...)
```

Arguments

object	A fitted dicca object (bi_projector with class "dicca")
newdata	New data matrix to predict from
...	Additional arguments (currently unused)

Details

This method projects new data to the latent space and applies the fitted temporal model (VAR for classic, ARIMA for compact variants).

Value

A list with components:

- scores - Observed latent scores
- scores_hat - Predicted latent scores

predict.dipca *Predict Method for DiPCA*

Description

Temporal forecasting using the VAR model fitted on latent scores.

Usage

```
## S3 method for class 'dipca'
predict(object, newdata, ...)
```

Arguments

object	A fitted dipca object (bi_projector with class "dipca")
newdata	New data matrix to predict from
...	Additional arguments (currently unused)

Details

This method projects new data to the latent space and applies the fitted VAR(s) model to generate one-step-ahead predictions. The first s time points have NA predictions since they require lagged values.

Value

A list with components:

- scores - Observed latent scores
- scores_hat - Predicted latent scores using VAR(s) model

predict.dipls *Predict Y using a fitted DiPLS model*

Description

Predict Y using a fitted DiPLS model

Usage

```
## S3 method for class 'dipls'
predict(object, newdata, ...)
```

Arguments

object	fitted "dipls" object.
newdata	matrix X or a list containing an element X.
...	unused.

Value

Matrix of one-step-ahead predictions for Y.

---

residuals.dicca *Residuals Method for DiCCA*

Description

Compute temporal prediction residuals for DiCCA model.

Usage

```
## S3 method for class 'dicca'
residuals(object, newdata = NULL, ...)
```

Arguments

object	A fitted dicca object (bi_projector with class "dicca")
newdata	New data matrix (optional, uses fitted data if missing)
...	Additional arguments (currently unused)

Details

Computes both latent score prediction errors from the VAR/ARIMA model and data-space reconstruction errors (dynamic whitening filter output). The first s time points have NA residuals since predictions require lagged values.

Value

A list with components:

- v - Score-space residuals ($T_t - \hat{T}_t$)
- e_hat - Data-space reconstruction residuals ($X_t - \hat{X}_t$)
- scores - Observed latent scores

residuals.dipca *Residuals Method for DiPCA*

Description

Compute temporal prediction residuals for DiPCA model.

Usage

```
## S3 method for class 'dipca'
residuals(object, newdata = NULL, ...)
```

Arguments

object	A fitted dipca object (bi_projector with class "dipca")
newdata	New data matrix (optional, uses fitted data if missing)
...	Additional arguments (currently unused)

Details

Computes both latent score prediction errors from the VAR model and data-space reconstruction errors. The first s time points have NA residuals since predictions require lagged values.

Value

A list with components:

- v - Score-space residuals ($T_t - \hat{T}_t$)
- e_hat - Data-space reconstruction residuals ($X_t - \hat{X}_t$)
- scores - Observed latent scores

residuals.dipls *Compute residuals for DiPLS model*

Description

Compute residuals for DiPLS model

Usage

```
## S3 method for class 'dipls'
residuals(object, newdata = NULL, ...)
```

Arguments

`object`	fitted `"dipls"` object.
`newdata`	optional list with elements `X` and `Y`. If NULL, uses training data (if stored).
`...`	unused.

Value

Matrix of residuals (Y - Yhat).

Index